

Amendments to the Claims

Please amend claims 1, 15, 18, 19, 23, 37, 40, 41, 44, 58, 61, 62, and 65, as shown below. All pending claims are reproduced below, including those that remain unchanged.

1. (Currently amended) A computer-implemented system to marshal and unmarshal data between XML and Java, comprising:
 - an XML data;
 - an XML schema which defines the XML data;
 - an XML type which is a Java type capable of accessing the XML data from within Java without mapping the XML data to an Java object; and
 - a compiler capable of generating the XML type from the XML schema.
2. (Original) The system according to claim 1, wherein:
 - the compiler is capable of generating the XML type based on the definition of a Java web services method.
3. (Original) The system according to claim 1, wherein:
 - the compiler is capable of generating the XML type based on a definition file.
4. (Original) The system according to claim 1, wherein:
 - the compiler is capable of compiling a Java project into one or more regular Java types.
5. (Original) The system according to claim 1, wherein:
 - the XML type can be a movable cursor, capable of reading anywhere within the XML data.
6. (Original) The system according to claim 1, wherein:
 - the XML type can be a immovable value, capable of referencing a fixed part of the XML data.

7. (Original) The system according to claim 1, wherein:
the XML type can be shared among multiple Java components.
8. (Original) The system according to claim 1, wherein:
the XML type is capable of updating the XML data within Java.
9. (Original) The system according to claim 1, wherein:
the XML type is capable of accessing and updating Java data using Java type methods.
10. (Original) The system according to claim 1, wherein:
the XML type is capable of accessing and updating a database.
11. (Original) The system according to claim 1, wherein:
the XML type is capable of a number of XML data operations, which include: querying XML data, transforming between XML types, and iterating over XML data document.
12. (Original) The system according to claim 1, further comprising:
an XML schema capable of defining the legal types of the XML data, which include constraints on data types and ranges of the data; and
constraints on the data types and ranges of the XML type.
13. (Original) The system according to claim 12, wherein:
the compiler is capable of generating constraints on the XML type from the XML schema on legal types of the XML data.
14. (Original) The system according to claim 12, wherein:
the constraints on the XML type are capable of validating the XML type.
15. (Currently amended) A system to transform types between XML and Java, comprising:
a Java type;

an XML type which is a Java type capable of accessing XML data from within Java without mapping the XML data to an Java object; and
an XML transformation capable of transforming a source type to a target type, wherein the source and target type can be either the XML type or the Java type.

16. (Original) The system according to claim 15, further comprising:
a global registry of XML transformations capable of looking up an existing XML transformation between a source and a target type.
17. (Original) The system according to claim 15, further comprising:
a library of XML transformations capable of looking up an existing XML transformation by name between a source and a target type.
18. (Currently amended) A system to marshal and unmarshal data between XML and Java, comprising:
an XML data;
a lightweight XML store capable of retaining the XML data as a searchable index; and an XML type which is a Java type capable of referencing the lightweight XML store and accessing the XML data from within Java without mapping the XML data to an Java object.
19. (Currently amended) A system to marshal and unmarshal data between XML and Java, comprising:
an XML data;
a lightweight XML store capable of retaining the XML data at the text or tag level; and an XML type which is a Java type capable of referencing the lightweight XML store and accessing the XML data from within Java without mapping the XML data to an Java object.

20. (Original) The system according to claim 19, wherein:

the lightweight XML store is capable of representing the retained XML data as a hierarchical structure.

21. (Original) The system according to claim 20, wherein:
the hierarchical structure can be a tree.
22. (Original) The system according to claim 19, wherein:
the XML type is capable of accessing the XML data incrementally.
23. (Currently amended) A method to marshal and unmarshal data between XML and Java, comprising:
defining an XML data using an XML schema; accessing the XML data via an XML type from within Java without mapping the XML data to an Java object; and
generating the XML type from the XML schema using a compiler.
24. (Original) The method according to claim 23, further comprising:
generating the XML type based on the definition of a Java web services method.
25. (Original) The method according to claim 23, further comprising:
generating the XML type based on a definition file.
26. (Original) The method according to claim 23, further comprising:
compiling a Java project into one or more regular Java types.
27. (Original) The method according to claim 23, further comprising:
utilizing the XML type as a movable cursor to read anywhere within the XML data.
28. (Original) The method according to claim 23, further comprising::
utilizing the XML type as an immovable value to reference a fixed part of the XML data

29. (Original) The method according to claim 23, further comprising:
sharing the XML type among multiple Java components.
30. (Original) The method according to claim 23, further comprising:
updating the XML data within Java via the XML type.
31. (Original) The method according to claim 23, further comprising:
accessing and updating Java data using Java type methods.
32. (Original) The method according to claim 23, further comprising:
accessing and updating a database via the XML type.
33. (Original) The method according to claim 23, further comprising:
utilizing a number of XML data operations via the XML type, these operations include:
querying XML data, transforming between XML types, and iterating over XML data
document.
34. (Original) The method according to claim 23, further comprising:
defining the legal types of the XML data via an XML schema, which include constraints
on data types and ranges of the XML data.
35. (Original) The method according to claim 34, further comprising:
generating constraints on the data types and ranges of the XML type from the XML
schema on legal types of the XML data.
36. (Original) The method according to claim 34, further comprising:
validating the XML type using the constraints on the XML type.
37. (Currently amended) A method to transform types between XML and Java, comprising:

utilizing a Java type;
utilizing an XML type which is a Java type capable of accessing an XML data from
within Java without mapping the XML data to an Java object; and
transforming a source type to a target type via an XML transformation, wherein the
source and target type can be either the XML type or the Java type.

38. (Original) The method according to claim 37, further comprising:
looking up an existing XML transformation between a source and a target type via a
global registry of XML transformations.
39. (Original) The method according to claim 37, further comprising:
looking up an existing XML transformation by name between a source and a target type
via a library of XML transformations.
40. (Currently amended) A method to marshal and unmarshal data between XML and Java,
comprising:
retaining an XML data as a searchable index via a lightweight XML store; and
referencing the lightweight XML store and accessing the XML data via the XML type
from within Java without first mapping the XML data to an Java object.
41. (Currently amended) A method to marshal and unmarshal data between XML and Java,
comprising:
retaining an XML data at the text or tag level via a lightweight XML store; and
referencing the lightweight XML store and accessing the XML data via the XML type
from within Java without first mapping the XML data to an Java object.
42. (Original) The method according to claim 41, further comprising:
representing the retained XML data as a hierarchical structure, which can be a tree.
43. (Original) The method according to claim 41, further comprising:

accessing the XML data incrementally via the XML type.

44. (Currently amended) A machine readable medium having instructions stored thereon that when executed by a processor cause a system to:

define an XML data using an XML schema;

access the XML data via an XML type from within Java without mapping the XML data to an Java object; and

generate the XML type from the XML schema using a compiler.

45. (Original) The machine readable medium of claim 44, further comprising instructions that when executed cause the system to:

generate the XML type based on the definition of a Java web services method.

46. (Original) The machine readable medium of claim 44, further comprising instructions that when executed cause the system to:

generate the XML type based on a definition file.

47. (Original) The machine readable medium of claim 44, further comprising instructions that when executed cause the system to:

compile a Java project into one or more regular Java types with the compiler.

48. (Original) The machine readable medium of claim 44, further comprising instructions that when executed cause the system to:

utilize the XML type as a movable cursor to read anywhere within the XML data.

49. (Original) The machine readable medium of claim 44, further comprising instructions that when executed cause the system to:

utilize the XML type as a immovable value to reference a fixed part of the XML data.

50. (Original) The machine readable medium of claim 44, further comprising instructions that when executed cause the system to:
- share the XML type among multiple Java components.
51. (Original) The machine readable medium of claim 44, further comprising instructions that when executed cause the system to:
- update the XML data within Java via the XML type.
52. (Original) The machine readable medium of claim 44, further comprising instructions that when executed cause the system to:
- access and update Java data using regular Java type methods.
53. (Original) The machine readable medium of claim 44, further comprising instructions that when executed cause the system to:
- access and update a database via the XML type.
54. (Original) The machine readable medium of claim 44, further comprising instructions that when executed cause the system to:
- utilize a number of XML data operations via the XML type, these operations include: querying XML data, transforming between XML types, and iterating over XML data document.
55. (Original) The machine readable medium of claim 44, further comprising instructions that when executed cause the system to:
- define the legal types of the XML data via an XML schema, which include constraints on data types and ranges of the XML data.
56. (Original) The machine readable medium of claim 55, further comprising instructions that when executed cause the system to:

generate constraints on the XML type from the XML schema on legal types of the XML data.

57. (Original) The machine readable medium of claim 55, further comprising instructions that when executed cause the system to:

validate the XML type using the constraints on the XML type.

58. (Currently amended) A machine readable medium having instructions stored thereon that when executed by a processor cause a system to:

utilize a Java type;

utilize an XML type which is a Java type capable of accessing an XML data from within Java without mapping the XML data to an Java object; and

transform a source type to a target type via an XML transformation, wherein the source and target type can be either the XML type or the Java type.

59. (Original) The machine readable medium of claim 58, further comprising instructions that when executed cause the system to:

look up an existing XML transformation between a source and a target type via a global registry of XML transformations.

60. (Original) The machine readable medium of claim 58, further comprising instructions that when executed cause the system to:

look up an existing XML transformation by name between a source and a target type via a library of XML transformations.

61. (Currently amended) A machine readable medium having instructions stored thereon that when executed by a processor cause a system to:

retain an XML data as a searchable index via a lightweight XML store; and

reference the lightweight XML store and access the XML data via the XML type from within Java without mapping the XML data to an Java object.

62. (Currently amended) A machine readable medium having instructions stored thereon that when executed by a processor cause a system to:

retain an XML data at the text or tag level via a lightweight XML store; and
reference the lightweight XML store and access the XML data via the XML type from within Java without mapping the XML data to an Java object.

63. (Original) The machine readable medium of claim 62, further comprising instructions that when executed cause the system to:

represent the retained XML data as a hierarchical structure, which can be a tree.

64. (Original) The machine readable medium of claim 62, further comprising instructions that when executed cause the system to:

access the XML data incrementally via the XML type.

65. (Currently amended) A system to marshal and unmarshal data between XML and Java, comprising:

means for defining an XML data using an XML schema;

means for accessing the XML data via an XML type from within Java without mapping the XML data to an Java object; and

means for generating the XML type from the XML schema using a compiler.

66. (Canceled) .